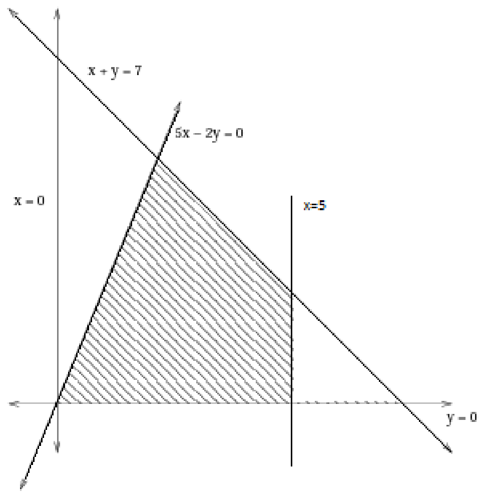
CMPSC 465 — Data Structures & Algorithms
Fall 2023 — Antonio Blanca and Yana Safonova

HW 13

1. (20 pts.) **Solving a Linear Program.**

(a) The feasible solution is the shaded region in the below graph:



(b) The new linear program uses a surplus variable a_1 for the first constraint and slack variables a_2 and a_3 for the second and third.

$$\begin{aligned}
\text{maximize}\quad & 5x+3y \\
\text{subject to}\quad & 5x-2y-a_1=0 \\
& x+y+a_2=7 \\
& x+a_3=5 \\
& x\geq 0 \\
& y\geq 0 \\
& a_1\geq 0 \\
& a_2\geq 0 \\
& a_3\geq 0
\end{aligned}$$

(c) The optimal solution is achieved in the upper right corner of the feasible region at the point $(5,2)$, and has a value of $5x+3y=31$. The initial tableau is shown below.

	x	y	a_1	a_2	a_3	b
s_1	5	-2	-1	0	0	0
s_2	1	1	0	1	0	7
s_3	1	0	0	0	1	5
z	-5	-3	0	0	0	0

Step 1: Because of the negative entry in column a_1, we first select the farthest left column with a positive entry in the row containing the -1 as the pivot column, which here is column x with a value

of 5. The smallest ratio of b column value to x column value is in row s_1 with a ratio of 0, so row s_1 is the pivot row. We now rescale that row so the 5 becomes 1 and perform elimination:

	x	y	a_1	a_2	a_3	b
s_1	1	-0.4	-0.2	0	0	0
s_2	0	1.4	0.2	1	0	7
s_3	0	0.4	0.2	0	1	5
z	0	-5	-1	0	0	0

Step 2: We now return to the normal pivot selection procedure. The column with the smallest value in row z is column y. The second pivot row is s_2 with the smallest ratio of 5. We now rescale that row so the 1.4 becomes 1 and perform elimination:

	x	y	a_1	a_2	a_3	b
s_1	1	0	-1/7	2/7	0	2
s_2	0	1	1/7	5/7	0	5
s_3	0	0	1/7	-2/7	1	3
z	0	0	-2/7	25/7	0	25

Step 3: The column with the smallest value in row z is column a_1. The third pivot row is s_3 because the other rows have already been used as pivots. We now rescale that row so the $1/7$ becomes 1 and perform elimination (here it's easier to eliminate first, then rescale):

	x	y	a_1	a_2	a_3	b
s_1	1	0	0	0	1	5
s_2	0	1	0	1	-1	2
s_3	0	0	1	-2	7	21
z	0	0	0	3	2	31

The maximum objective function value is the entry in row z, column b: 31. The assignment that produces that optimal value are the b values corresponding to the pivots (cells of value 1) in columns x and y: $x = 5$ and $y = 2$.

(d) The dual linear program is:

$$\begin{aligned}
\text{minimize} \quad & 7b + 5c \\
\text{subject to} \quad & 5a + b + c \geq 5 \\
& -2a + b \geq 3 \\
& a \leq 0 \\
& b \geq 0 \\
& c \geq 0
\end{aligned}$$

This is found by multiplying the first three constraints (the constraints that are for more than just non-negativity) by a, b, and c, respectively. Note that $a \leq 0$ comes from the fact that the first constraint uses $\geq$ instead of $\leq$ in the primal. Note that the counts of variables and interesting constraints flipped from 2 variables with 3 constraints in the primal to 3 variables with 2 constraints in the dual.

The dual linear program is minimized at $a = 0$, $b = 3$, $c = 2$, with a value of 31. Therefore, the optimal values of the primal and dual objective functions are equal, and strong duality is confirmed to hold for this problem (as it does for all linear programs).

2. (20 pts.) **Spaceship.** Let x_1 be the number of oxidizer units provided for compartment 1, and define similarly x_2 for compartment 2 and x_3 for compartment 3. The probability that all units fail in compartment

1 is $(0.3)^{x_1}$, for compartment 2 is $(0.4)^{x_2}$ and for compartment 3 is $(0.2)^{x_3}$. The probability that all units fail in all compartments is $(0.3)^{x_1}(0.4)^{x_2}(0.2)^{x_3}$. The exponential function is non-linear, so we take logarithm (which preserves ordering of numbers) to get the linear program

$$\text{Minimize } \log(0.3)x_1 + \log(0.4)x_2 + \log(0.2)x_3$$
$$\text{subject to } \begin{cases} 40x_1 + 50x_2 + 30x_3 \le 500 & \text{space constraint (cu in.)} \\ 15x_1 + 20x_2 + 10x_3 \le 200 & \text{weight constraint (lb)} \\ 30x_1 + 35x_2 + 25x_3 \le 400 & \text{cost constraint (\$1000)} \\ \log(0.3)x_1, \log(0.4)x_2, \log(0.2)x_3 \le \log(0.05) & \text{reliability constraints (log scale)} \end{cases}.$$

(Note that the reliability constraints imply the non-negativity constraints that $x_1, x_2, x_3 \ge 0$.)

3. (20 pts.) **Maximum Flow.**

(a) Let $f_{u,v}$ be the flow on edge $e = (u, v)$. We can write the linear program as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{(s,v)\in E} f_{s,v} \\ \text{subject to} \quad & \sum_{(u,v)\in E} f_{u,v} = \sum_{(v,w)\in E} f_{v,w} \quad \forall v \in V - \{s,t\} \quad \text{(flow conservation)} \\ & f_{u,v} \le c_{u,v} \quad \forall (u,v) \in E \quad \text{(capacity constraint)} \\ & f_{u,v} \ge 0 \quad \forall (u,v) \in E \end{aligned}$$

(b) Let f_p be the flow through s-t path p. Let the set of all s-t paths be P.

$$\begin{aligned} \text{maximize} \quad & \sum_{p\in P} f_p \\ \text{subject to} \quad & \sum_{p\in P:(u,v)\in p} f_p \le c_{u,v} \quad \forall (u,v) \in E \quad \text{(capacity constraint)} \\ & f_p \ge 0 \quad \forall p \in P \end{aligned}$$

Note that the flow conservation constraint does not need to be specified explicitly when we consider entire paths at a time instead of individual edges.

(c) We multiply each of the capacity constraints by a dual variable $y_{u,v}$ ($|E|$ variables total).

$$\begin{aligned} \text{minimize} \quad & \sum_{(u,v)\in E} c_{u,v} y_{u,v} \\ \text{subject to} \quad & \sum_{(u,v)\in p} y_{u,v} \ge 1 \quad \forall p \in P \\ & y_{u,v} \ge 0 \quad \forall (u,v) \in E \end{aligned}$$

These dual variables can be interpreted as a length between s and t. The first constraint ensures s and t are separated by the end, just like they are in the residual graph after running Ford-Fulkerson.

(d) The capacity across the cut is the sum of the capacity of the edges that cross the cut in the direction from A to B. We need to construct a feasible solution to the dual with an objective function equal to that sum. The easiest way is to set $y_{u,v} = 1$ if $u \in A$ and $v \in B$, otherwise $y_{u,v} = 0$. Because every s-t path must cross the cut, the first constraint is satisfied. This assignment also makes the objective function equal to the capacity of the cut described at the beginning of the solution, as only edges crossing the cut in the correct direction have their capacities counted.

4. (20 pts.) **Cheap Max 3-SAT.**

(a) Given an instance of this problem, along with a proposed assignment that claims to satisfy at least a clauses by using at most b variables set to true, a polynomial-time certifier can verify the following:

(1) There are indeed at most b variables set to true (with a linear pass over the assignment).

(2) There are indeed at least a satisfied clauses (with a linear pass over the clauses, computing the value of each under the given assignment).

Both checks can be computed in polynomial time, and the certificate (the proposed assignment) takes up polynomial space as it assigns a single value to each of the variables. Therefore this problem can be verified in polynomial time, and it is in **NP**.

(b) Polynomial reduction from 3-SAT. Given an instance of the 3-SAT problem (a Boolean 3-CNF formula with n variables and m clauses), we can construct an instance of the Cheap Max 3-SAT problem as follows:

- Keep the Boolean formula the same.
- Set $a = m$, ensuring all clauses are satisfied.
- Set $b = n$, because we don't care how many variables we have to set to true.

Solving this instance of the Cheap Max 3-SAT will tell us whether the original 3-SAT instance is satisfiable. If it is satisfiable, the same assignment that satisfies the Cheap Max 3-SAT problem will also satisfy the original 3-SAT problem. This reduction is polynomial time because only the variables a and b must be set, a constant-time operation. Therefore, 3-SAT reduces to Cheap Max 3-SAT in polynomial time, and this problem is **NP**-Hard.

5. (20 pts.) **Hitting Set.**

(a) Given an instance of this problem, along with a proposed subset H that claims to hit every S_i while containing at most k elements, a polynomial-time certifier can verify the following:

(1) The proposed H is indeed size at most size k (by counting the elements in linear time).

(2) The proposed H hits each of the S_i. This can be done with a linear scan over each S_i asking "Is this item in H?". Even implemented with $O(n)$ searches, this is upper bounded by $|H|\sum_i |S_i| = O(|U|\sum_i |S_i|)$, which is polynomial on the input size since each S_i must be part of the input.

Both checks can be made in polynomial time, and the proposed H cannot be larger than U (even if it was, the first check would simply reject it). Therefore, this problem can be verified in polynomial time, and it is in **NP**.

(b) Reduction from Vertex Cover. Given an instance of Vertex Cover (an undirected graph $G = (V,E)$ and an integer k), we create an instance of Hitting Set as follows:

- Let $U = V$, the set of all vertices.
- Let $S_i = \{u,v\}$ for all $(u,v) \in E$. In other words, create a subset for each edge in the graph. Hitting a subset is analogous to covering an edge.
- Keep k the same.

Solving this Hitting Set instance also solves the Vertex Cover instance. Notice how, if every S_i is hit by some $H \subseteq U$, every edge would be covered by choosing the same subset of V in the graph. There is a one-to-one correspondence between vertices and items and between edges and subsets. Creating the hitting set instance takes polynomial time, as the inputs are largely the same. Retrieving the solution to the Vertex Cover from the Hitting Set is similarly easy, simply choose the vertices corresponding to items chosen in H. Therefore, Vertex Cover reduces to Hitting Set in polynomial time, and Hittitng Set is **NP**-Hard.

Rubric:

Problem 1, 20 pts

(a) 4 pts for correct plot

(b) 4 pts for correct linear program

(c) 6 pts: 2 for initial tableau, 1 per correct tableau step, 1 for correct answer.

(d) 6 pts: 4 for correct dual, 2 for the optimal variable values.

Problem 2, 20 pts
4 pts for the objective function, 4 pts per constraint.

Problem 3, 20 pts

(a) 5 pts for correct LP.

(b) 5 pts for correct LP.

(c) 5 pts for correct dual.

(d) 5 pts for correct variable assignment.

Problem 4, 20 pts

(a) 8 pts for NP argument (may be shorter, make sure they say what must be verified and how it's polynomial time).

(b) 12 pts for reduction (could be a different reduction, make sure they explain how it's polynomial time).

Problem 5, 20 pts

(a) 8 pts for NP argument (may be shorter, make sure they say what must be verified and how it's polynomial time).

(b) 12 pts for reduction (could be a different reduction, make sure they explain how it's polynomial time).